Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1997 Proceedings

Americas Conference on Information Systems (AMCIS)

8-15-1997

DATABASE AUDITING

Levent V. Orman
Cornell University, Orman@johnson.cornell.edu

DATABASE AUDITING

Levent V. Orman
Cornell University
Malott Hall
Ithaca, NY 14853-4201
orman@johnson.cornell.edu

Database management systems are the primary tools of automated bookkeeping [1, 7, 16, 23]. They are used to store and maintain a record of all business transactions, and also create and maintain accounting data that are derived from those raw transactions. As in manual bookkeeping systems, automated bookkeeping systems have to be monitored carefully to maintain data quality. A large variety of errors can seep into organizational databases, ranging from data entry errors to violations of accounting standards. In fact, there is considerable evidence documenting the poor quality of data in organizational databases [12, 19, 21]. It appears that although database systems have revolutionized bookkeeping in terms of efficiency and speed, the ability to detect errors and maintain quality has not kept pace.

The primary tool of database auditing is database integrity constraints [7, 8, 16, 23]. They are logical statements that capture the semantics of data and the intent of the transactions. They have to be satisfied by the database at all times for the data to be correct and complete. They can be incorporated into the database through the data dictionary facilities, or implemented as external views [4, 7, 9, 23]. Data dictionaries are constructs designed to capture a description of the database, its valid states, and correct transformations resulting from update transactions. They often contain integrity constraints that capture the basic semantics of data using devices such as keys and functional relationships. However, these constraints are often very limited in scope and usually for human manual analysis rather than automatic enforcement. In their most general form, integrity constraints can be maintained and enforced as external views that contain integrity violations. These external views can be considered queries that are executed on a continuous basis and return the data that cause violations of integrity. There is one external view for each database constraint, and they are expected to be empty so long as the database is in a correct state. They are monitored by the database management system periodically, or even continuously, to ensure that they are empty [11, 17]. A continuous integrity monitoring and enforcement system appears more desirable than a periodic system, but there are serious implementation problems. A continuous, real-time integrity enforcement system requires the execution of all relevant integrity constraints after each transaction that updates the database, to ensure that no violations have resulted from the transaction. Few commercial systems can provide this level of service due to the high cost of executing integrity constraints. Integrity constraints are complex queries that are executed against very large databases, and executing a large number of them after each transaction is a very costly undertaking. In fact, a real-time integrity enforcement system for a large, highly active database can easily overwhelm the system resources and bring it to a halt, or at least cause major unacceptable delays in the processing of transactions [13, 15, 17, 20]. Such delays themselves are significant sources of error in commercial databases due to obsolescence of data, making it impossible to

provide an error-free database. All commercial systems are forced to tolerate significant levels of errors in their databases, and finding the optimum level is one objective of this article. The evidence suggests that most commercial databases have very high error rates [12, 19, 24]. Some actually have very little or no integrity enforcement, rendering the database potentially unreliable. More importantly, most commercial database systems provide no measurement and evaluation of errors and error rates for an effective statistical quality control system for data [2, 3, 10, 14, 18].

There are three major approaches to database auditing and integrity enforcement. The first is periodic enforcement where all integrity constraints are executed at the end of a fixed period, and the errors and violations that have occurred during the period are caught and corrected. The errors occur at random points in time during the period, and they are carried in the system until the end of the period. Clearly, during the period, any queries and reports involving the erroneous data will produce erroneous responses. The determination of the optimum period is critical to this strategy to minimize the error rates in the stored data. The optimum period may be different for each integrity constraint depending on a variety of parameters such as the execution cost of the constraint, and the error rates in the relevant incoming transactions. Clearly, high execution cost of a constraint will lead to a longer optimum period, and larger error rates in the database will have to be tolerated. Similarly, larger error rates in incoming transactions will lead to a smaller optimum period, and larger execution costs for integrity constraints will have to be tolerated.

The second approach is a transaction counting approach where the constraints are executed after every n transactions. A transaction is an atomic update operation which inserts, deletes, or modifies one record in the database. As in the previous approach, the optimum selection of the number of transactions n is critical to the efficiency of this approach, and to the minimization of error rates in the stored data. A special case of the counting approach is the real-time continuous integrity enforcement where the integrity constraints are executed after each transaction (n=1). A real-time continuous integrity enforcement strategy is often considered ideal in the literature since it presumably allows no errors in the system, albeit at great cost. We will show in section 4 that real-time continuous integrity enforcement is rarely the optimum strategy even when we restrict the objective to the minimization of error rates in the stored data. Very frequent execution of the integrity constraints may actually increase the error rates in the stored data due to delays it causes in executing transactions, and the consequent delays in keeping the database up-to-date. Moreover, the strategy is often unfeasible due to the high execution cost of integrity constraints as discussed in section 1.

The third approach is a variation on the real-time continuous integrity enforcement. Under this approach integrity constraints are executed after each transaction, but not in their full and complete form. A simplified version of each integrity constraint is executed to test if a new error was introduced into the database by a given transaction, assuming that the database was error free before the transaction [13, 17]. The restriction of the constraint enforcement to a specific transaction leads to considerable simplification of integrity constraints and actually makes a real-time continuous integrity enforcement a

feasible strategy. However, the simplification comes at a very high price: the execution of all relevant integrity constraints after each transaction albeit in their simplified form, since simplified constraints are only valid one-transaction at a time. Such frequent execution of integrity constraints may actually increase the average error rates in the stored data due to delays it causes in executing transactions and updating the database, as discussed above.

The major tradeoff is between correctness and timeliness of data. Both may contribute to the error rates in the database: one by entering incorrect data into the system, the other by failure to update the data and hence causing it to lapse into obsolescence. Either occurrence will be counted as an error, and the objective throughout this article will be to minimize the error rates in the stored data, whether it is caused by the entry of new erroneous data or by the failure to update the existing data. The failure to update is commonly caused by the queuing of transactions as they wait for the processing of previous transactions, and the execution of integrity constraints. The processing delays are exacerbated by the locking of files during the execution of integrity constraints to prevent interference from transactions. The possibility of interference between transactions and integrity constraints is demonstrated by the following example.

Example 2.1: Consider a bank with savings and checking accounts, a customer with $100 in each account, a transaction by that customer transferring $100 from one account to the other, and a constraint imposed on the bank database that requires each customer to maintain $200 total in their combined savings and checking accounts. The transaction has two major steps: T1 = withdrawal of $100 from one account, T2 = deposit of $100 into the second account. The integrity enforcement has three steps: I1 = retrieval of the account balance from the first account, I2 = retrieval of the account balance from the second account, I3 = testing of the total ³ 200. If the transaction and the integrity enforcement procedure are allowed to run concurrently, they can interfere with each otherís operations and cause unpredictable results. Clearly, there is no violation of integrity by this transaction since it merely transfers money from one account to the other, but if the steps are executed in such a sequence to interfere with each other, a violation may be indicated. Consider the sequence (T1, I1, I2, T2, T3) which will indicate a violation since T1 will reduce one account balance by $100, and I1 and I2 will receive account balances 0 and $100 with a total of $100 which is clearly less than $200, i.e., a violation of the constraint. On the other hand, the sequences (T1, T2, I1, I2, I3) or (T1, I1, T2, I2, I3) will both execute the integrity constraint correctly and cause no violations.

Clearly, update transactions can interfere with the execution of integrity constraints if they are run concurrently. Consequently, during the execution of integrity constraints, the relevant files are locked and no new updates are allowed to prevent interference between the integrity enforcement procedure and the changing state of the database. The database queries and report requests during this period can be answered, but only using a snapshot of the database as it existed before the start of the enforcement procedure. New update transactions arriving during the integrity enforcement are queued, and the system is updated after the integrity enforcement is complete. Consequently, during the time of integrity enforcement, the database becomes stale due to delays in processing the new

updates. This is a clear disadvantage of very frequent integrity enforcement, and it has to be balanced against the cost of carrying errors in the system due to infrequent integrity enforcement. The objective in this article is to determine the optimum balance between these costs.

BIBLIOGRAPHY

1. Amer, T., A.D. Bailey, P. De. A review of the computer information systems research related to accounting and auditing. *Journal of Information Systems*. 3-28, Fall 1987.
2. Ballou, D.P., H.L. Pazer. Designing information systems to optimize the accuracy-timeliness tradeoff. *Information Systems Research*. 1995.
3. Ballou, D.P., H.L. Pazer. Modeling data and process quality in multi-input multi-output information systems. *Management Science* 31, 2, 150-162, 1985.
4. Cerullo, M.J. General controls in computer systems. *Computers and Security*. 4, 33-45, 1985.
5. Cinlar, E. *Introduction to Stochastic Processes*. Prentice Hall 1975.
6. Cooper, R.B. *Introduction to Queuing Theory.* Ceepress Books 1990.
7. Date, C.J. *An Introduction to Database Systems*. Addison Wesley 1995.
8. Davis, G.B., R. Weber. The impact of advanced computer systems on controls and audit procedures: A theory and empirical test. *Auditing: A Journal of Practice and Theory*. Spring 1986.
9. Groomer, S.M., U.S. Murthy. Continuous auditing of database applications: An embedded audit module approach. *Journal of Information Systems*. 53-69, Spring 1989.
10. Hansen, J.V., W.F. Messier. A relational approach to decision support for EDP auditing. *Communications of ACM*. 1129-1133, November 1984.
11. Koch, H.S. Auditing on-line systems: An evaluation of parallel versus continuous and intermittent simulation. *Computers and Security*. 3, 9-19, 1984.
12. Laudon, K.C. Data quality and due process in large interorganizational record systems. *Communications of ACM*. 29, 1, 4-11, 1986.
13. McCune, W., L. Henschen. Maintaining state constraints in relational databases. *Journal of ACM*. 36, 1, 1989.
14. Morey, R.C. Estimating and improving the quality of information in MIS. *Communications of ACM*. 25, 5, 337-342, 1982.
15. Motro, A. Integrity = validity + completeness. *ACM Transactions on Database Systems*. 14, 4, 480-502, 1989.
16. Orman, L.V. Functional semantics of accounting data. *Journal of Accounting Systems*. 1, 2, 6-29, 1990.
17. Orman, L.V. Constraint by example. Decision Sport Systems 17,1,3-12,1996.
18. Paradise, D.G., W.L. Fuerst. An MIS data quality methodology based on optimal error detection. *Journal of Information Systems*. 5, 1, 48-66, 1991.
19. Redman, T.L. *Data Quality: Management and Technology*. Bantom Books 1992.
20. Segev A., J.L. Zhao. Rule Management in expert database systems. *Management Science*. 40, 6, 685-707, 1994.

21. Wang, R.Y., V.C. Storey, C.P. Firth. A framework for the analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering*. 7, 4, 623-639, 1995.
22. Wand, Y., R. Weber. A model of control and audit procedure change in evolving data processing systems. *Journal of Accounting Research*. 11, 3, 273-295, 1989.
23. Weber, R. EDP Auditing: Conceptual Foundations and Practices. Prentice Hall 1988.